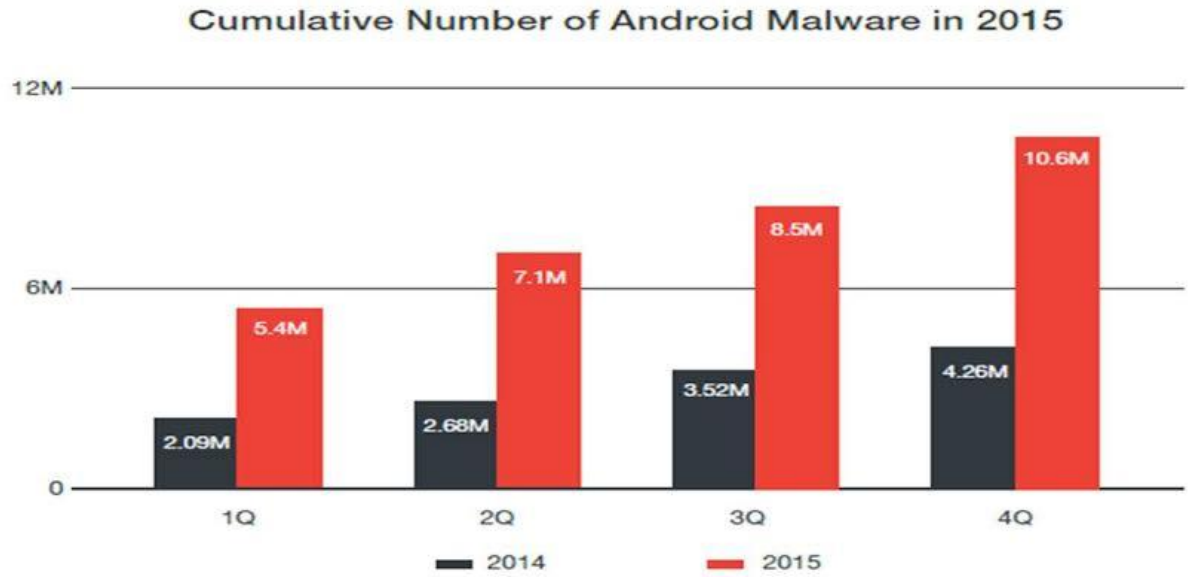
# Malware detection from app permissions using ML

Suma Kasa, 130050075
Vegini Eadula, 130050048
Dinesh Kota, 120050051
Rakesh Ranjan Nayak, 120050047
Krishna Deepak M, 120050057

# Android Market

- Android market is the fastest growing mobile application platform
- The openness of the Android market attracts both benign and malicious developers
- Android also allows installing third-party applications that may increase the spread of Android malware
- So, the smartphone has become a hotbed for malwares

# Beware of Malware




*Compared to 2014 data, the number of Android malware doubled by the end of 2015.*

# Android Application Model

- Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps
- An app can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more
- Each app has access only to the components that it requires to do its work and no more

# Android Components

- **Activities**
  - An *activity* represents a single screen with a user interface.
- **Services**
  - A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes.
- **Content providers**
  - A *content provider* manages a shared set of app data.
- **Broadcast Receivers**
  - A *broadcast receiver* is a component that responds to system-wide broadcast announcements.

# Intents

- Activities, services, and broadcast receivers—are activated by an asynchronous message called as *intent*.
- An intent is created with an `Intent` object, which defines a message to activate either a specific component or a specific *type* of component—an intent can be either explicit or implicit, respectively.
- Example:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
startActivity(sendIntent);
```

# Manifest file

An Android manifest file declares the following things for an application:

- App components
- App permissions
- Minimum API level
- Hardware and software features used
- API libraries the app needs to be linked with

# Android Security Model

- Android security model highly relies on permission-based mechanism.
- There are about 130 permissions that govern access to different resources in the Android system.
- Generally, users are with rare knowledge to determine if permissions might be harmful or not.
- Malwares generally take advantage of the scenario and prompt for more sensitive permissions than they actually need to.
- Ex. READ_CONTACTS, MASTER_CLEAR

# Analysis of Android applications

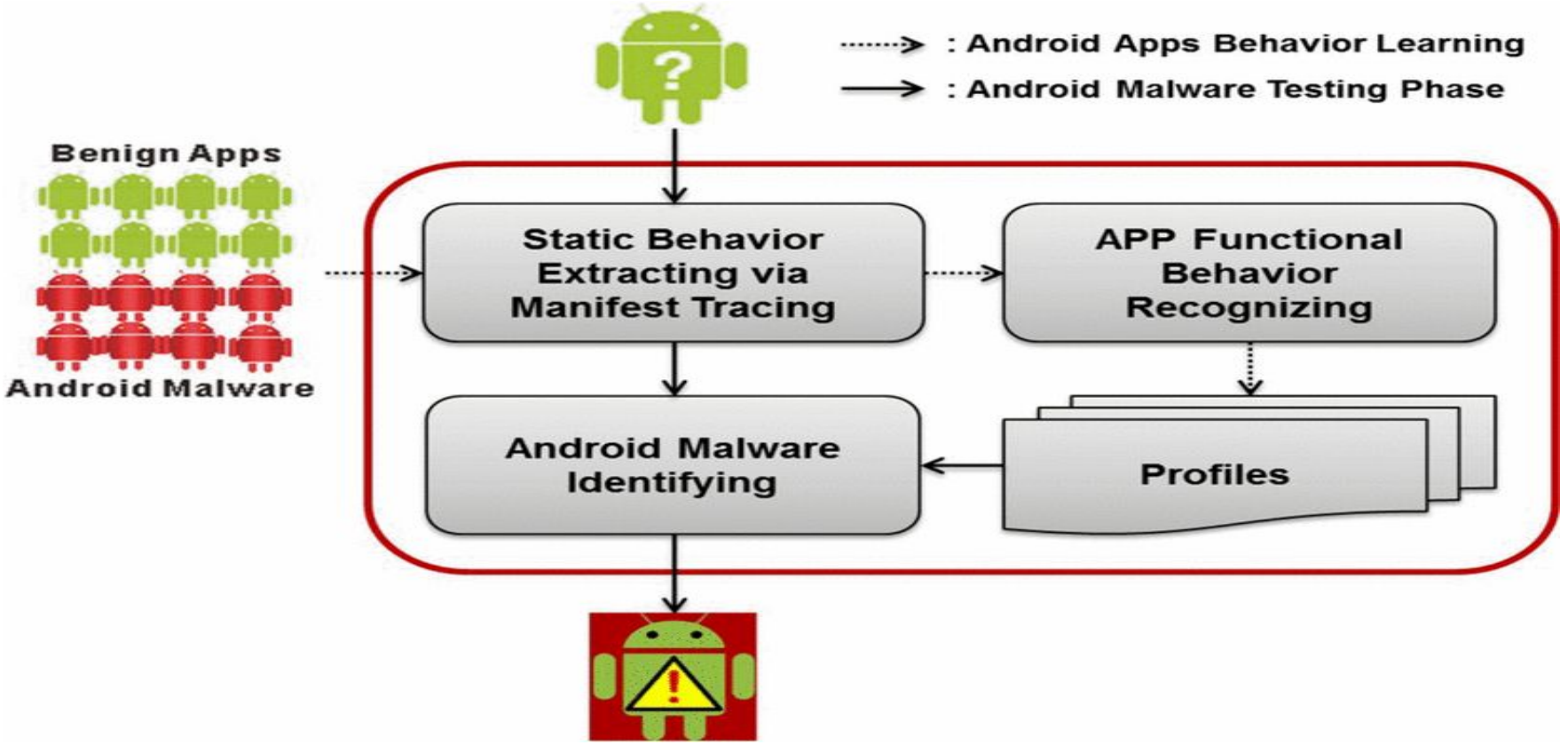
Dynamic and static analysis are two main approaches to analyse applications codes

| Static Analysis | Dynamic Analysis |
| --- | --- |
| ● Useful information about the app is extracted by analysing the manifest files and the bytecodes<br>● Might not provide a wholesome view (Can't analyze system calls)<br>● Low cost | ● The app is run in a sandbox environment and detail profiles of RAM usage, system calls are analysed<br>● Can provide a very comprehensive view<br>● Has a high cost because of environment deployment and manual efforts |

# Our Approach

1. Data collection - Collecting benign and malicious apps
2. Use *apktool* to decompile apks into source files
3. Manifest parsing to get permission and component informations
4. Static Behavior Extraction
5. Detecting feature vectors
6. Training our system using ML algos
7. Classification of an application to Malware or not

# Our Approach



# Datasets

- For benign applications, we downloaded application from google play store. We tested the applications in existing malware detection tools to ensure that they don’t contain any malware.
- For malicious apps, we used “andromalshare” and “contagiominidump” sites which seemed to be the only two sites offering malwares for testing purpose
- For our project, we used a dataset comprising of 70 benign applications and 50 malicious applications
- We ensure that the 70 benign applications belong to a wide range of categories

# Feature Vectors used

- Permissions
  - Set of permissions
  - Number of permissions
- Components Information
  - No. of activities, services and broadcast receivers
- Intent Information
  - Type of intent filters
  - Data passed through intents inside components

# ML algos used on feature vectors

- We use SVD(Singular Value Decomposition) to find out the optimal number of clusters in the low rank approximation
  - We used 95% spectral density to find out the value of l.
- We used K-Means clustering algorithm to cluster our dataset into l clusters and kNN algorithm to find out which cluster the test application belongs to
- From other application feature vectors from the same cluster, we classify if the test application is malicious or not using perceptron.

# Results

- We trained our system on 70 benign and 50 malicious applications.
- We used 4 malicious and 3 benign application to test our system and we got correct classification.
- Demo Video links:
  - https://www.dropbox.com/s/o20uqtlfodm00l3/video1_redbus_benign.mov?dl=0
  - https://www.dropbox.com/s/tawky0zvxncc042/video2_malicious_app.mov?dl=0

# Paper implemented

**DroidMat: Android Malware Detection through Manifest and API Calls Tracing,** Wu, Dong-Jie; Mao, Ching-Hao; Wei, Te-En; Lee, Hahn-Ming; Wu, Kuo-Ping, Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on , Issue Date: 9-10 Aug. 2012

- Special thanks to **Penfei Cao** and **Mila Parkour** for providing us sample malwares for our project